



# FREMetar: Efficient and Flexible Metadata Rewriting

Kyle Olivo
Aparna Radhakrishnan
V. Balaji
Serguei Nikonov
GFDL

GO-ESSP 9th Workshop
May 10th - 11th
Asheville, NC

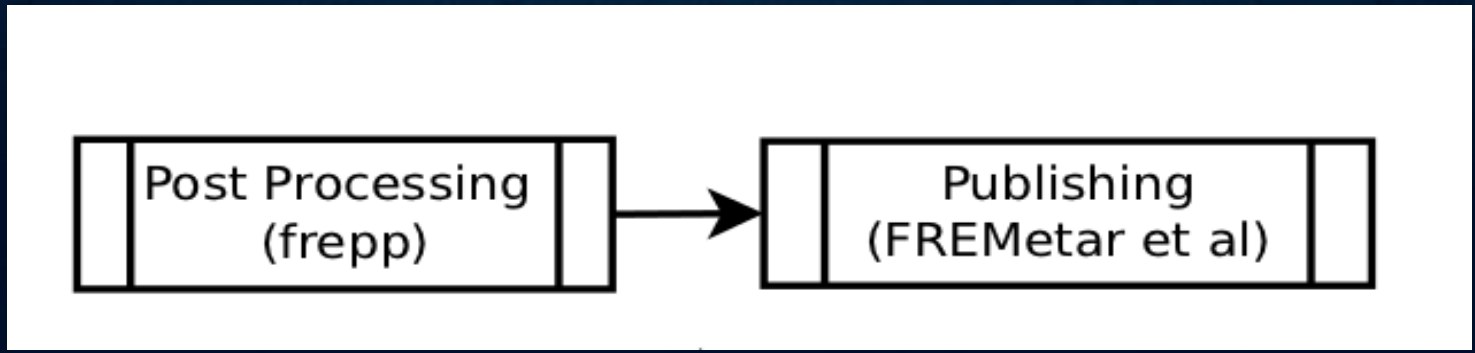
# What is it?

- GFDL analogue to CMOR (Climate Model Output Rewriter)
- Originally operated on metadata only
- Natural extension of existing tools

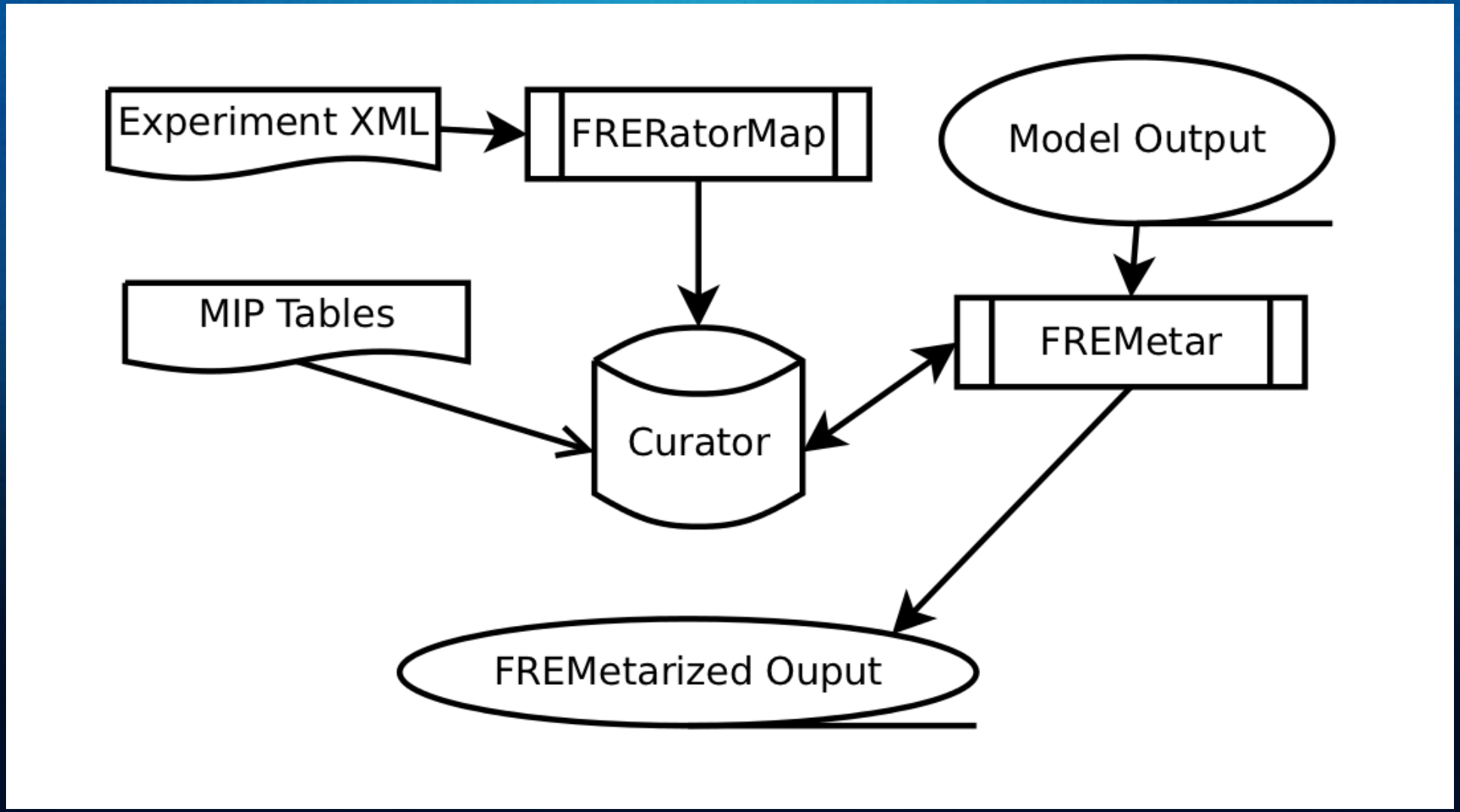
```
netcdf wfo_Omon_GFDL-ESM2M_piControl_r1i1p1_049601-050012 { // format variant: 64bit
dimensions:
        rlon = 360 ;
        rlat = 200 ;
        time = UNLIMITED ; // (60 currently)
        bnds = 2 ;
        vertices = 4 ;
variables:
        double rlon(rlon) ;
                rlon:long_name = "longitude in rotated pole grid" ;
                rlon:units = "degrees" ;
                rlon:axis = "X" ;
                rlon:standard_name = "grid_longitude" ;
        double rlat(rlat) ;
                rlat:long_name = "latitude in rotated pole grid" ;
                rlat:units = "degrees" ;
```

# What are the benefits?

- CMOR rewrites all data as well as metadata -- computationally costly
- Easier integration with post processing software (frepp) and other tools (FREratorMap)
- Centralization of all metadata information, automatic use of existing experiment XML information



# What is the architecture?



# What is stored in the database?

- Experiment details

| | exper_id | inner_name | outer_name | project_id |
|---|---|---|---|---|
| 1 | exper_id_7fm6cI8IJk | ESM2M_pi-control_C1 | pre-industrial | ipcc_ar5 |

- Project details
  - AR5, ACCMIP, CLIVAR, etc.

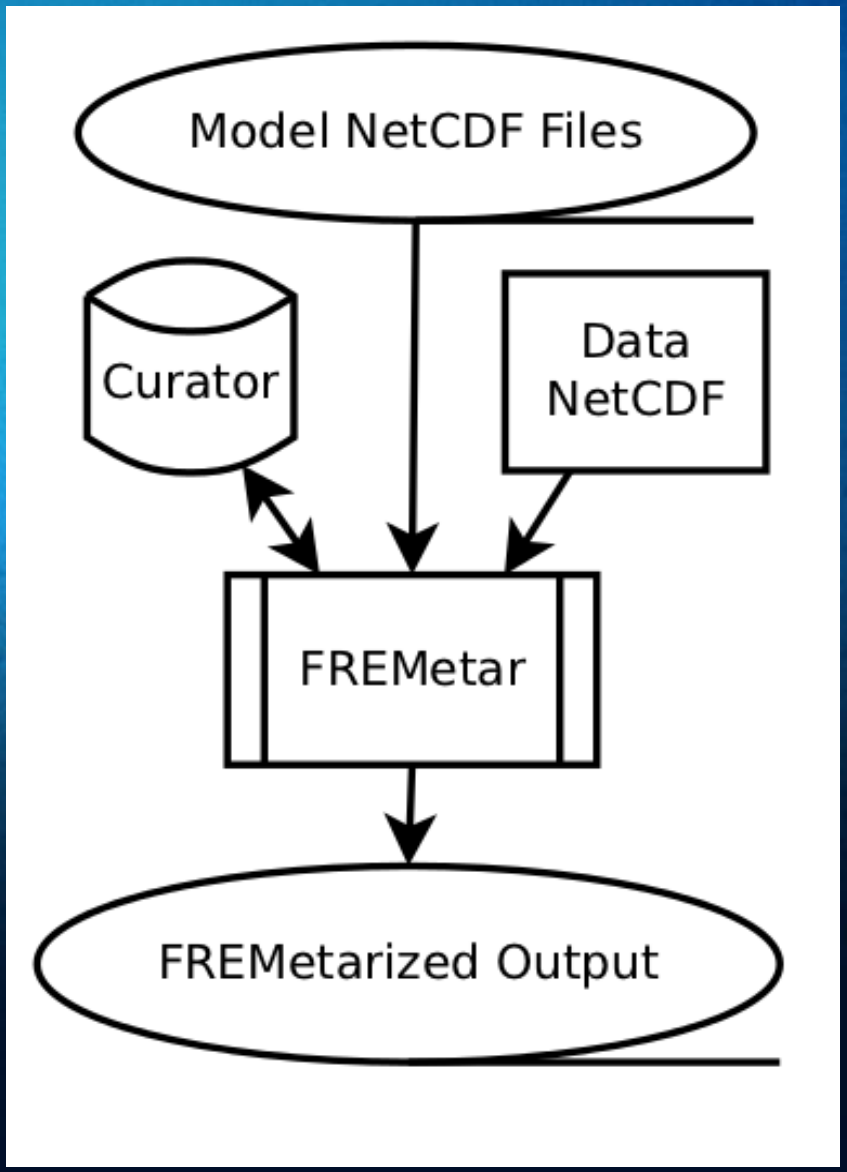
- Variable details

| | project_id | var_id | long_name | units | original_units | inner_varname | proj_varname |
|---|---|---|---|---|---|---|---|
| 1 | ipcc_ar5 | 3 | Near-Surface Air | K | deg_k | t_ref | tas |

- Project metadata
  - Institution, product, conventions, etc.

# How is it designed?

- Written in Java
- MySQL
- Java NetCDF API
- NetCDF Operators
- Auxiliary NetCDF file (provides data regarding grid details, auxiliary coordinate variables, etc)



# How is it used?

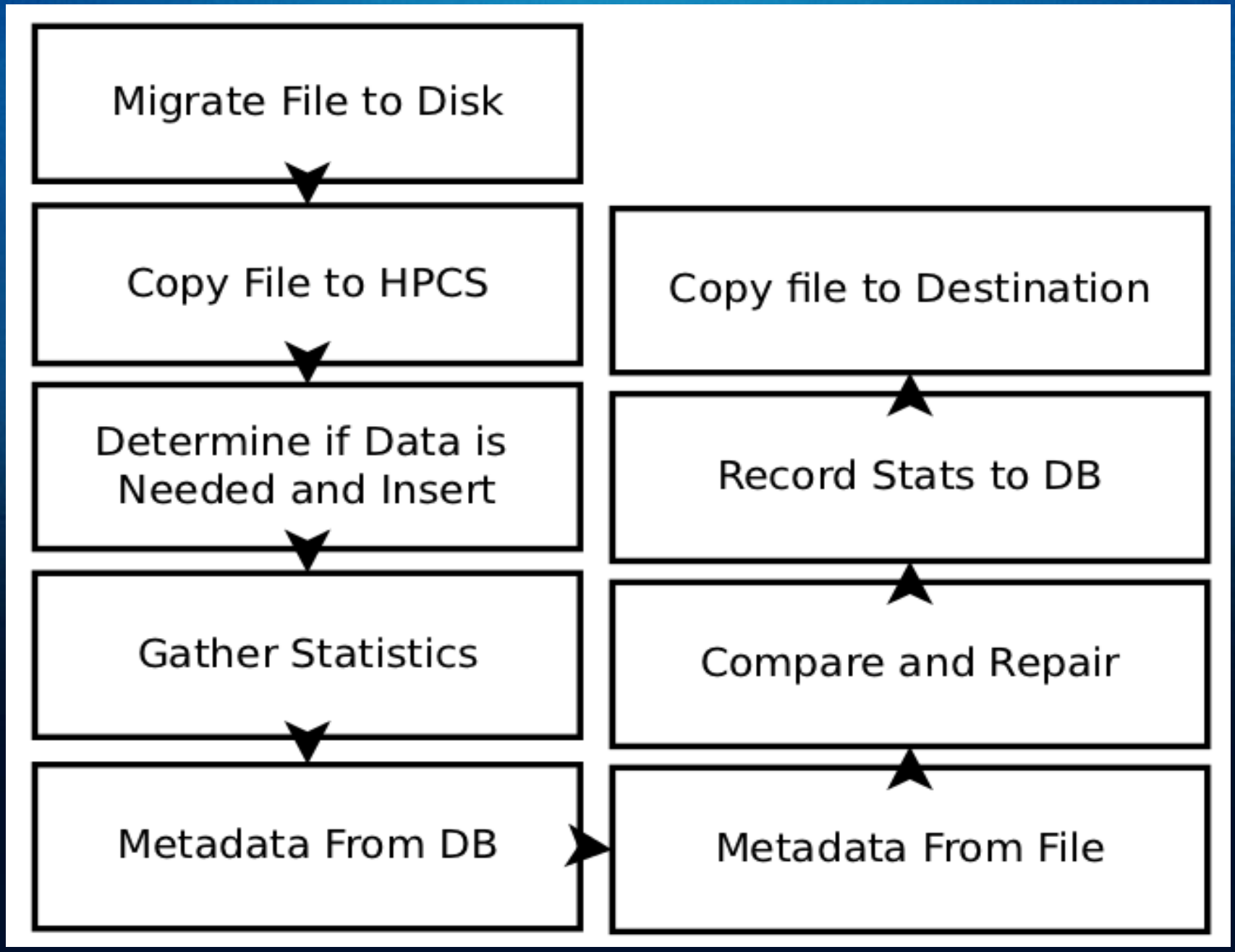
```
frametar -d <dir> | --directory <dir> -e <exper_id> | --experid <exper_id>
      -z <realiz_id> | --realizid <realiz_id> -r <run_id> | --runid <run_id>

      Optional Parameters:
      [ -o <output dir> | --output <output dir> ] [ -c <configfile> | --config <configfile> ]
      [ -h | --help ] [ -v | --verbose ]  [ -p | --preview ] [ -q | --quiet ] [ -a | --diag ]
      [ -s <db server> | --server <db server> ] [ -n <db name> | --name <db name> ] [ -f ]
```

Synopsis:

| | |
|---|---|
| -h | --help |
| -v | --verbose |
| -d <dir> | --directory <dir> |
| -e <exper_id> | --experid <exper_id> |
| -z <realiz_id> | --realizid <realiz_id> |
| -r <run_id> | --runid <run_id> |
| -t <t start> | --time <t start> |
| -k | --chunk <years> |
| | |
| -o <out dir> | --output <out dir> |
| -c <configfile> | --config <configfile> |
| -i | --versioning |
| | |
| -p | --preview |
| -a | --diag |
| -q | --quiet |
| -m <v1,v2..> | --variable <v1,v2..> |
| -y <t1,t2..> | --table <t1,t2..> |
| -w | |
| -s <dbserver> | --server <dbserver> |
| -n <dbname> | --name <dbname> |
| -f | |

# How does it operate?



# How does it support multiple standards?

- FREMetar is project independent
- Insert experiment XML
- Insert project specific metadata details (from MIP table)
- Create variable mappings
- Create proper metadata entries

# What can we conclude?

- FREMetar utilizes existing strengths in GFDL's publishing architecture
- We should automate use of existing information repositories (MIP tables)
- The database centric workflow is powerful and flexible
- We can (and will) make the pipeline even stronger for future projects and the next assessment

Thanks!